SMAT: Synchronous Multimedia and Annotation Tool

Michelle Potts Steves, M. Ranganathan, Emile Morse
National Institute of Standards and Technology
{msteves, mranga, emile.morse} @nist.gov

Abstract

We describe the design and use of SMAT (Synchronous Multimedia and Annotation Tool), a tool designed to be part of a scientific collaboratory for use in a robotic, arc-welding research project at the National Institute of Standards and Technology (NIST). The primary functional requirements of SMAT are to provide the capability to capture, synchronize, play back, and annotate multimedia data in a multi-platform, distributed environment. To meet these requirements, SMAT was designed as a control and integration framework that exploits existing tools to render specific media types and control annotation sessions. SMAT defines a component architecture framework where existing tools can be plugged in and controlled using a distributed, event-driven, tool-bus architecture. SMAT's modular architecture enables control inputs to come from anywhere in the distributed collaborative environment, thus allowing for simultaneous remote and local control of the tool, as well as painless interfacing with the existing collaborative environment. SMAT is built on an agent middleware called AGNI (Agents at NIST), also developed at NIST. We give an overview of AGNI that can be used to build failure-resilient, distributed, event-driven applications. In addition to describing SMAT's design, interface and underlying middleware, we present performance information, an initial analysis of welding users' experiences and feedback, related work, and directions for further SMAT development.

Introduction

The increasing globalization of manufacturing and distribution of enterprises demands concurrent information exchange and collaboration throughout the product development life cycle. This creates an increasing dependence on information technology to share disparate data among geographically dispersed staff. Globalization trends and recent advances in information technology (IT) provide an opportunity now for computer supported cooperative work (CSCW) [11]. Imagine this scenario, where advanced, integrated, CSCW technology is used to enable the efficient trouble-shooting of a manufacturing process problem by one of the few experts available in a highly specialized field:

Jade, a welding engineer, is reading her e-mail, when a yellow, flashing icon beeps on her computer. She opens up the icon and sees a message about trouble on a welding line as well as a hyperlink to a virtual collaboration space. She clicks the hyperlink and sees a long list of good welds denoted by a green color code, in a dynamically updated data table. Additionally, there are two welds that are color coded yellow – the second suspect yellow weld automatically triggered the warning icon on her computer desktop. Clicking the yellow icon invokes a Virtual Reality Modeling Language (VRML) current-voltage graph plotted over the geometry of the weld, overlaid with a transparent template for a good weld, with tolerance ranges indicated. The first bad weld shows a problem at the beginning of the weld sequence but the second shows a problem toward the end.

Jade starts a multimedia playback for each of the welds that includes audio, video, sensor, and controller information. She associates some spikes in the graphs with some sounds and visual signs in the welds themselves that make her suspect a faulty power supply. She contacts Harry, the job setter for the problem welding cell, and asks him to use his PC to join her in the virtual collaboration space. She points out features she sees in the data, and together they decide to call in an electrician to check out the power supply. Jade suggests that Harry show the electrician the current-voltage graphs to help explain the problem they suspect with the power supply. Jade makes some annotations in the welding data and sends an email to her European counterpart as a "heads-up".

The basic technology components exist for enabling this and other types of collaboration that will be common in future global manufacturing environments. The challenge is in understanding the collaboration requirements, and identifying and integrating appropriate collaboration technology solutions. In this paper, we focus on a component of a scientific collaboratory developed to support geographically dispersed, manufacturing, trouble-shooting processes. We describe the design of SMAT, its

environment and underpinnings, benefits, and future development directions.

Background

Successful groupware deployment is more than installing video conferencing on every computer workstation available; it is a thoughtful exercise that considers many factors. Specifically, its aim is to apply the right collaborative tool(s) for a job given many factors. For instance, there is considerable research showing that adding audio to desktop conferencing improves problem-solving among team members; however, there typically is no benefit to adding video [21]. Meanwhile, new research shows promise for video providing significant benefit when used in tasks involving speakers with different priorities and different linguistic capabilities [21]. This type of finding holds the promise to support increasingly diverse work teams in our increasingly global economy. With the aid of CSCW, groupware will eventually support the way we work, i.e., cooperatively, and often distantly located.

In the introduction, we described a scenario where collaboration tools are used to trouble-shoot a problem with an automated welding process. There are many similar scenarios repeated throughout manufacturing process engineering and other domains, where collaborative tools would be useful. In these scenarios, there are problems with a process and there are various types of "experts" including operators and engineers who are not necessarily co-located. They need to communicate regarding the problem, symptoms, past history, and so on; and, they need to suggest and experiment with alternatives (e.g., using integrated simulation tools).

Welding

Researchers at NIST's Manufacturing Engineering Laboratory and Information Technology Laboratory are in the process of instituting and assessing collaboration technologies for manufacturing applications [19]. We are particularly interested in how collaboration tools can be used effectively in manufacturing environments and how manufacturing practices will change as a result of their use. We expect our studies will yield useful insights into future, data-interchange-standards needs, as well as advance the state of the art and practice in CSCW deployment for manufacturing and possibly other domains. We are employing user-centered design, as it has been shown to increase the likelihood of acceptance, effectiveness, and user satisfaction of IT systems [e.g., 8, 9, 10]. Field studies are being used to document the work and show where there are changes in manufacturing processes and data exchange requirements as a result of these systems' use.

Our current work, set in the context of automated, gas-metal, robotic welding, assesses the deployment and use of collaboration technologies for process engineering and trouble-shooting. In industry, there is a relative scarcity of welding engineers; this shortage of engineers causes delays in a variety of activities such as configuring new welding lines and trouble-shooting problem welds. One component of a solution to this problem is to use welding engineers' time more productively. Currently, time is wasted in travel and engineers' not being able to oversee problems at co-located sites concurrently because of inadequate tools. If a virtual presence could be established, some, if not substantial increases in productivity could be achieved. Collaboration is a vital component in the testing and trouble-shooting of automated, robotic welding equipment, welding processes, and the analysis of subsequent welds by a welding team. Collaboration technology holds the promise of realizing substantial savings in productivity by allowing geographically dispersed welding teams to trouble-shoot bad welds over time and distance, as conceptualized in the "Jade" illustration. (To address the issue of whether the NIST collaboratory addresses real-world problems, these ideas were presented and greeted with unanimous support at the National Advanced Manufacturing Testbed (NAMT) Gas-Metal Arc Welding Workshop, September 1998.)

NIST welding researchers have a similar collaboration scenario, where, as a geographically dispersed team, they are working to define interface standards between welding work cell components, controllers, and power supplies. A functioning welding testbed has been implemented for testing the interfaces between components, controllers, and power supplies. Analysis of welds is performed to verify effective operation [16]. Just as in the industrial operations scenario, task appropriate collaboration and data visualization technologies hold the promise of effective collaboration over time and distance.

Summarized requirements

The welding collaboratory requirements were gathered and documented, and are summarized here:

1. Weld analysis requires collaboration among participants in distant locations and time zones. Asynchronous communications are required.
2. The welding process generates data in various formats that multiple people need to access, review, and annotate. Not all formats have been specified to date.
3. NIST researchers require a central repository of data, which supports appropriate access permission controls, supports heterogeneous data formats, and allows for organizing data and interactions around a central principle, e.g., around a particular weld or part.

4. Engineers need to divide time among several problems, and therefore do not want the burden of being in lockstep synchrony with each current problem.
5. High networking bandwidth solutions can not be imposed because some welding industries and sites do not have high capacity networking infrastructure.
6. Potential solutions must run on the major computing platforms.
7. To analyze welds, a data visualization tool incorporating an overlay of bad welds on a good weld template with delineated tolerance ranges is needed.
8. To identify trends and analyze problems, a visualization of a time series of good and bad welds per work cell is needed.
9. A synchronized replay of weld audio, video, sensor, and controller data is needed. Further, the capability to make annotations at notable events during the weld data replay is especially important.

A combination of collaborative tools is being used since no single, commercial off-the-shelf (COTS) tool was available that met all the requirements. To accelerate tool deployment, a COTS tool, Teamwave Workplace[1] (TW), was chosen as the foundation tool for the welding collaboratory, as it met many of the requirements. It was extensible, and was developed in an environment supporting CSCW research. SMAT was developed in-house and integrated with Teamwave Workplace for a seamless collaboratory. While SMAT specifically addresses requirement #9, it also addresses requirements 1-6. An overview of the collaboratory systems follows.

1. Certain commercial products are identified in this document for the purpose of evaluating a class of collaboration technologies. This identification does not imply any recommendation or endorsement by the National Institute of Standards and Technology.

The NIST Collaboratory

The NIST collaboratory is comprised of a number of systems, depicted in Figure 1. The welding system itself has a number of modules. However, for the purposes of this paper, it can be viewed as a remote instrument that has software controls, but requires human intervention to run (fixturing, control program generation, etc.). Images can be captured in the welding facility by manual interaction with a pan/tilt/zoom camera that is controlled from a web page and is mounted above the welding cell. In some cases multiple images of a single weld are obtained after the weld is completed, since lighting conditions do not usually allow image capture during actual welding. Data files are saved to a networked file store.

A daemon process waits for the creation of new welding files and processes them. The daemon copies the files, massages the data to synchronize the streams, creates SMAT-readable files, places the pertinent files in a file transfer protocol (FTP) repository, builds a metadata file

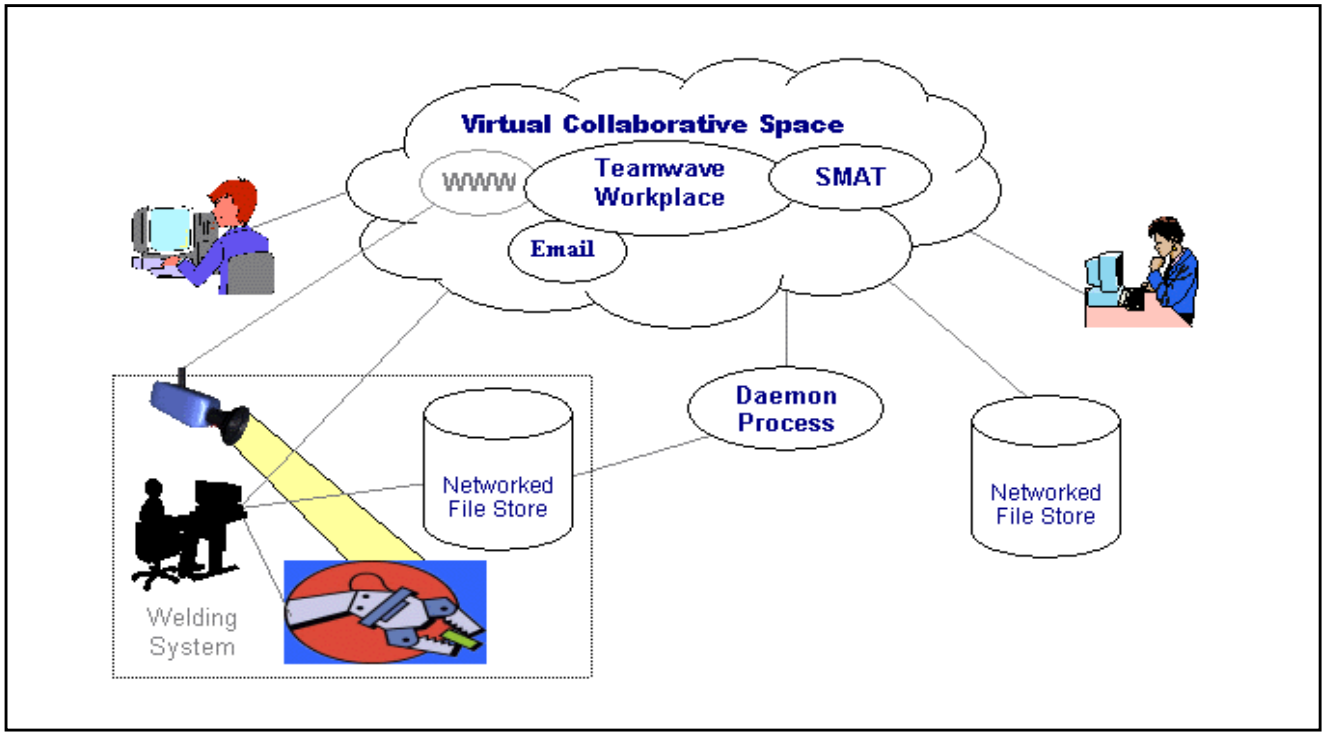

Figure 1: NIST Manufacturing Collaboratory

for SMAT that consolidates pointers to the various related data streams, and puts a pointer to the metadata file in a predetermined location in Teamwave Workplace. Team members can then access the data through the Teamwave Workplace interface or a standalone version of SMAT. Security of the data is managed through the password protection mechanism built into FTP.

Teamwave Workplace (TW) [17] is used as the overarching collaboration tool in the collaboratory. TW is a rooms-based, collaboration system with a whiteboard backdrop. Rooms provide boundaries for data groupings and user interactions as well as a metaphor for easing the transition to groupware [7]. Team members control the composition and organization of data within rooms, in how they organize various tools housing their data, such as file viewers, file holders, PostIt™ notes, and message boards. The system provides for synchronous and asynchronous user interactions, but importantly, these interactions are in the context of relevant data. The tool set is extensible, e.g., custom tools such as SMAT can be added. Figure 2 shows a screen shot of a room in Teamwave Workplace supporting the analysis activities of a weld. The left-most portion of the room shows summary status and navigation information, the center and right portions show data for a good weld and a bad weld, respectively. At the bottom of the window is an in-progress chat session regarding the analysis of the latest weld data.

Figure 2: Teamwave Workplace "room" showing various tools and data for two related welds

SMAT is a software solution for capturing, synchronizing, playing back, and annotating multimedia data streams in a multi-platform (Windows NT and UNIX), distributed environment. SMAT annotations support collaboration. We anticipate extending SMAT's capabilities to include synchronous playback by various team members. Further, SMAT provides the unusual capability of playing multiple synchronized data steams of heterogeneous formats at the same time. SMAT is both a visualization tool for simultaneously viewing any number of streams of heterogeneous multimedia data, and a collaboration tool, providing an annotation facility for notable "events" during the viewing of those streams.

Related Work

Annotations are an important concept in single-user systems [18] and have been adapted for use in collaborative systems. The utility of linking a person's thoughts to material being viewed is critical to making sense of decisions that were made based on the original material. Annotations provide a method for managing corporate memory and justifying decisions.

There are numerous, recent examples of annotation software in the literature. Some systems support personal annotations much as a person taking notes at a meeting. Other systems focus on 'secretarial' annotations, e.g., one person recording and sharing the results with a group. Still other systems treat annotations of novel media such as video and audio. SMAT derives its uniqueness by addressing all of these issues in a shared context – multiple authors and multiple readers (re)viewing and annotating multiple, heterogeneous synchronized data streams.

The Classroom 2000 project at Georgia Tech [1, 2, 3] incorporates some of the same ideas as those used by SMAT. The educational context supports teachers making alterations (annotations) to materials written on an electronic whiteboard. Students can make similar annotations in their personal notes. Sharing of notes between teachers and students is a more difficult issue that has been tackled in the most recent paper [13]. Their approach via linking is similar to the text annotations provided by SMAT.

Video and multimodal annotations have been studied in [4, 6]. Written and spoken annotations have been tested for their ability to support indexing and search through multimodal archives. Although SMAT currently does not present video or audio information, its architecture and design philosophy make it critical to consider incorporating enhanced annotation modes to deal with enriched formats.

We derived some experience from our exploratory work in extending the Synchronized Multimedia Integration Language (SMIL) to support annotations [20]. This work

centered around a tool called ACTS (Annotation Collaboration Tool via SMIL), that allows users to create and modify annotations along the replay of synchronized multimedia streams. From our prototypical work with ACTS, we found that current SMIL functionality met some, but not all, of our requirements. However, SMIL is evolving, and we anticipate the next generation of SMIL (SMIL 2.0), that is expected to provide event handlers, hierarchical layout, tighter timing and synchronization, etc., will better support and meet our requirements.

SMAT Functional Scenario

Sensors in various parts of the welding system and welding cell controller produce data of various media types – for example, video, audio, discrete images and discretely sampled current and voltage. The primary functional requirements for SMAT are to provide the capability to synchronize and play back the captured multimedia data after the weld is complete, and to provide a means to stop the play back and make an annotation at any point in time. After the annotation is composed, it can be made available for other users to view and, if they wish, annotate the annotation. During subsequent sessions, the media and the annotations are played back in synchronous fashion. During replay, annotations appear at appropriate locations along the timeline.

SMAT Design and Implementation

To meet these requirements, SMAT was designed as a control and integration framework that exploits existing tools to play specific media types. Each tool to be controlled exports an Application Protocol Interface (API) or mechanism, such as Component Object Model (COM) [5], that permits it to be controlled from another process. The tools are tied together using a common control bus. The idea of this bus is much the same as the idea of a bus in computer hardware. Components are tied together by plugging them into the software bus in the same fashion as cards are plugged into a hardware bus. The components in this case are slave processes that play the different multimedia files and take commands from the bus. One of our goals was to make SMAT operational on multiple computing platforms. Unfortunately, there is no uniformity in component architectures across platforms. In order to achieve cross-platform uniformity, the interfaces to the tools under control must be made uniform, which we accomplish by wrapping a controller script around each tool. In general, each tool may have its own idiosyncrasies for external communication. We encapsulate these via a software driver wrapper that hides the communication complexities from the control layer and registers standardized callbacks with the control layer. The callbacks include a ‘start’ interface, a ‘stop’ interface, a ‘quit’ interface, a ‘timer tick’ interface and a ‘seek’ interface, all of which are called from the controller at appropriate times. It is up to the driver to communicate with the slave tool if necessary on each of these calls. To enhance usability, we use Tk [12] window embedding to achieve a uniform look and feel. Each tool that has an embeddable top-level window is embedded in a common canvas. The architecture is shown in Figure 3.

The key innovation here is separation of control from the tool that is being controlled. By architecting and building tools in this fashion, control can be distributed and modularized. For example, the global controller in Figure 3 can receive commands from anywhere in the distributed environment. This makes it possible to distribute the control and enable synchronous as well as asynchronous collaboration. For example, a user may enable another user to control the tool by simply enabling the global controller to accept control inputs from the other user. This local-remote transparency and distribution of control formed the requirements of our distributed scripting environment that can be used for building a variety of collaborative tools including the synchronous multimedia annotation tool that is the subject of this paper. We considered the event abstractions to be of general interest and hence built a distributed, event-oriented scripting environment called AGNI, that is the subject of the next section.

AGNI Overview

Collaborative environments have a common set of communication and infrastructure requirements that can be abstracted into a framework. These requirements are as follows:

1. Event-oriented structure: A user may perform an action at his or her workstation and the effects of that action have to be fielded by the other participants (either synchronously or asynchronously) for collaboration to occur.
2. Distribution: Clearly, the primary requirement for any collaborative environment is that it be distributed (i.e. be able to run across multiple sites).
3. Security: Unless we are working in a closed environment, it is necessary to incorporate security mechanisms into the system to prevent inadvertent or malicious disruptions of the environment.
4. Failure Detection and Recovery: Failures are a common occurrence in distributed systems and the collaborative environment should be able to detect these and support appropriate recovery. A primary requirement for failure recovery is that the failure state be well defined so that appropriate failure handlers can be incorporated.

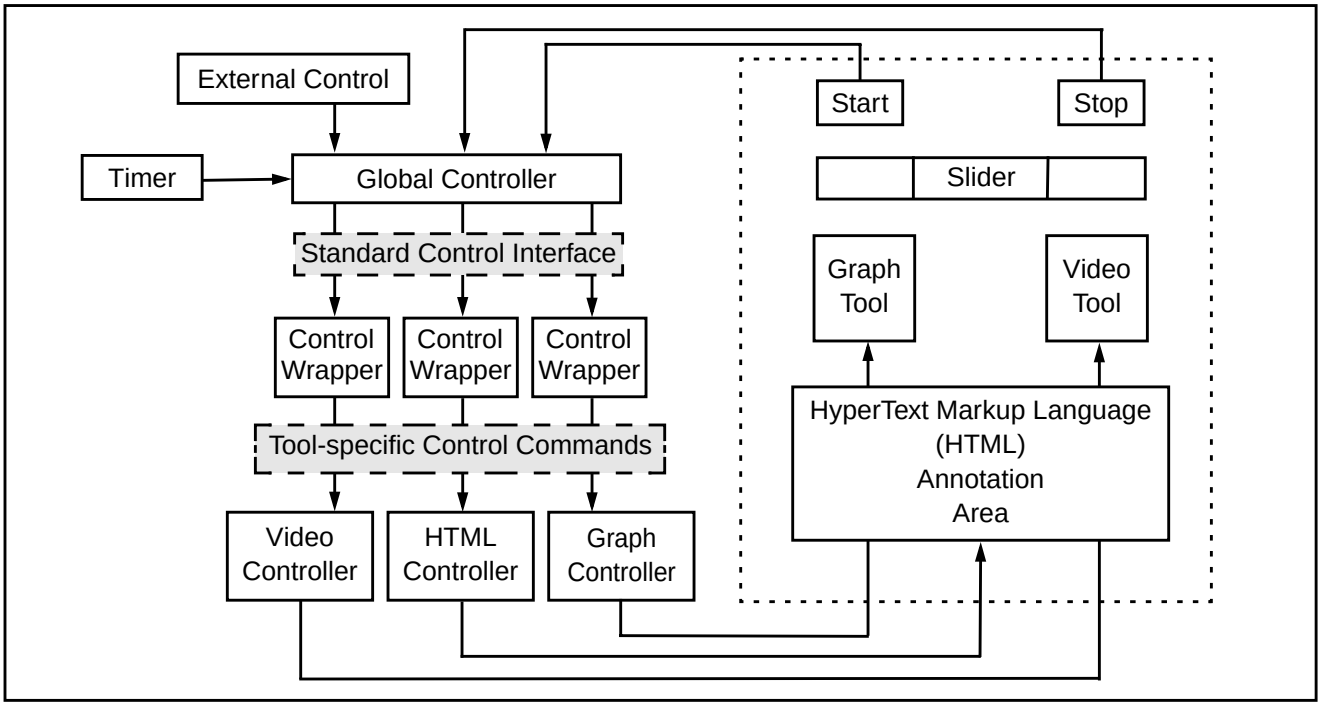

Figure 3: The SMAT Architecture

5. Heterogeneity: Given the plethora of existing tools and technologies, a pragmatic approach to building collaborative tools aims to combine tools into more powerful tools in much the same way as we have composed SMAT.

Given these common requirements, we designed and constructed a Tcl/Tk-based [12], distributed, scripting tool, called AGNI (Agents at NIST, also, Sanskrit for "fire"). The basic idea is to enable the logical design of a distributed, collaborative environment independent of the physical placement of components. Using this approach, the designer may design composite, event-oriented tools and distribute the control by pushing the control elements to the collaborating parties' workstations. Stated differently, our approach is to generalize the basic idea of an applet to a distributed system. The basic abstractions in AGNI are Mobile Streams, Sites and Sessions.

A *Mobile Stream* (MStream) is a named communication end-point in a distributed system that can be moved from machine to machine, (1) while a distributed computation is in-progress, and (2) while maintaining a pre-defined ordering guarantee of message consumption with respect to the order in which messages are sent to it. An MStream has a globally unique name.

We refer to any processor that supports an MStream execution environment as a *Site*. The closest analogy to an MStream is a mobile, active mailbox. MStreams provide a first-in-first-out (FIFO) message-ordering guarantee. While mailboxes are usually stationary, MStreams, have the ability to move from Site to Site dynamically. While mailboxes are usually passive, message arrival at an MStream can trigger the concurrent execution of message-consumption event Handlers registered with the MStream.

A distributed system consists of one or more Sites. A collection of Sites participating in a distributed application is called a *Session*. Each Session has a designated, trusted, reliable Site called a *Session Leader*. Each Site is assigned a *Location Identifier* that uniquely identifies it within a given Session. New Sites may be added and removed from the Session at any time. An MStream may be located on, or moved to, any Site in the Session that allows it to reside there. MStreams may be opened like sockets and messages may be appended to them. *Multiple Event Handlers* (Handlers) may be dynamically attached to and detached from an MStream. Handlers are invoked on discrete changes in system state such as message delivery, MStream relocations, new Handler attachments, new Site additions and Site failures. We refer to these discrete changes in system state as *Events*. Handlers are attached by *Agents* that provide an execution environment and thread for the Handlers that they attach. That is, an Agent specifies a collection of Handlers that all use the same thread of execution and interpreter. Logically, a distributed system constructed using AGNI is structured as shown in Figure 4.

Handlers can communicate with each other by appending messages to MStreams. These messages are delivered asynchronously to the registered append Handlers in the same order that they were issued.

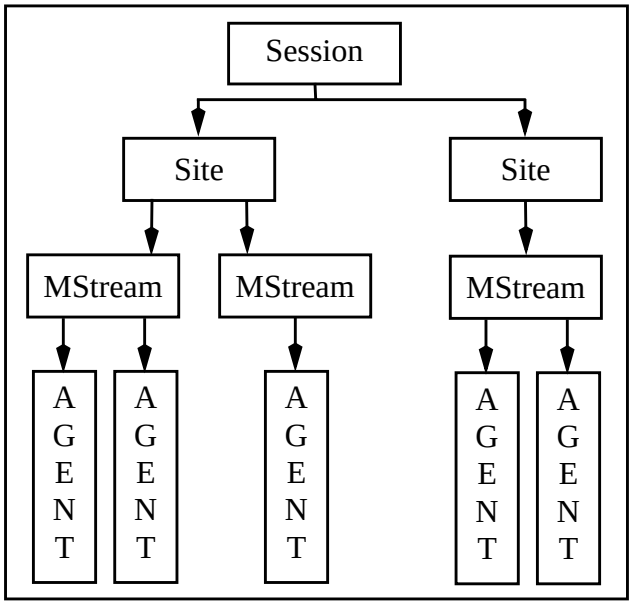

Figure 4: A logical view of a distributed system constructed using AGNI.

(Synchronous delivery of messages is supported as an option, but asynchronous delivery is expected to be the common case.) By asynchronous delivery we mean that the sender does not block until the message has been consumed in order to continue its execution. The underlying messaging code takes care of message buffering and reliable delivery.

An application built using the AGNI middleware, may be thought of as consisting of two distinct parts – an active part and a reactive part. The reactive part consists of MStreams and Handlers. The active part or *Shell* lives outside the middleware and drives it. A Shell may connect to the middleware and issue requests and may exit at any time. The reactive part is persistent for the life of the Session. Figure 5 shows an example of a distributed application that consists of two MStreams. When a message is consumed, the 'on_stream_append' Handler runs. When a message is sent to the MStream 'bar', it relocates itself and prints its current location.

AGNI allows mobility and dynamic extensibility by permitting MStream movement while there are pending, undelivered messages. Provided the state of the distributed application can be encapsulated and represented as strings, it can be stored in a briefcase structure, moved along with the MStream, and re-instantiated at a new Site. This allows personal mobility in a collaborative environment. In order to preserve message ordering in the presence of such mobility, we have designed a custom communication protocol on top of user datagram protocol (UDP) that preserves order in the presence of mobility and failure [15].

A centralized, reliable *Failure Manager* handles failures. This is a reliable location where the failure

```
stream_create foo
stream_create bar
register_agent foo {} {
    stream_open bar
    on_stream_append {
        stream_append bar $argv
    }
}
register_agent bar {} {
    on_stream_append {
        puts $argv
        stream_relocate 1
    }
    on_stream_relocation {
        puts "I am at [stream_location]"
    }
}
```

Figure 5: An example script that realizes a self-reconfiguring event-driven distributed application.

handler that is attached to an MStream executes. The failure recovery mechanism also resynchronizes sequence numbers so that message delivery ordering can be preserved. The design of AGNI is described in greater detail in [14].

Applications built using Mobile Streams can be extended from multiple points of control; any Handler or Shell that has acquired an open MStream handle can attempt to re-configure or extend the reactive part of the system and these actions can occur concurrently. While this adds great flexibility, it also raises several security and stability issues. We provide a means of restricting system reconfiguration and extension using control Events that can invoke policy Handlers. Only privileged Agents may register these policy Handlers. We follow a discretionary control philosophy by providing just the mechanism and leaving the policy up to individual applications. Controls may be placed via policy Handlers at the session-wide level, the site-wide level, and with individual MStreams for various security-relevant Events.

Graphical User Interface

The user interface for the SMAT application is shown in Figure 6. SMAT is written in Tcl/Tk. Graphical user interface (GUI) events are also sent through the control bus. The elements of the display are configurable through a script file containing the names, locations, and types of the component files. In the configuration shown in Figure 6, there are three graphs, an image and a text widget. The left and center graphs represent two data streams captured at the time a weld was made. The right graph shows data that was generated by applying an algorithm to these two data streams, along with some constant data based on the

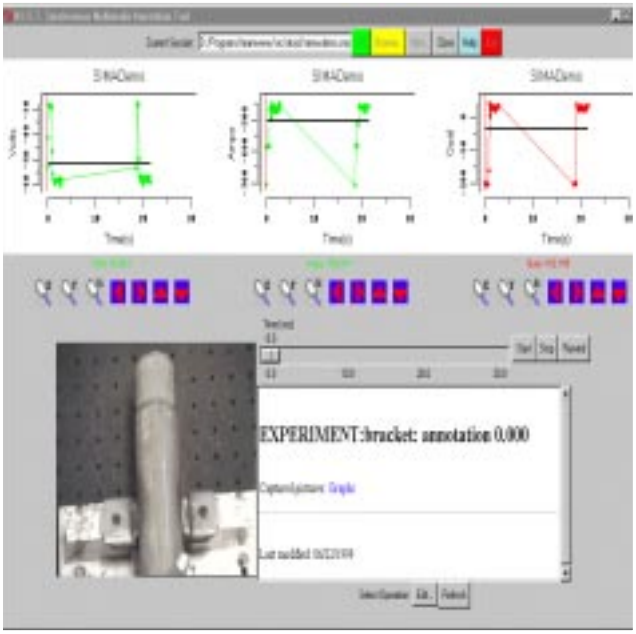

Figure 6: User Interface of SMAT

conditions under which the weld was made. Each data stream is time-stamped. Zooming and translation of the graphs is controlled through the series of icons shown just below each graph.

The user interacts with the graphs and images by using the replay feature, which is located immediately above the text box. The user controls the progress of the replay by choosing ‘Start’, ‘Stop’ or ‘Rewind’. In addition, the associated slider may be dragged to any position of the time scale. During replay, a vertical line on each of the graphs moves synchronously through the graph. Simultaneously, the most recent image and most recent annotation are updated in their respective locations.

Annotation facility

SMAT not only allows visual review of a weld and its associated parameters, but also provides the user with the ability to create time-based annotations. If the replay feature is paused, selecting the ‘Edit’ button at the bottom of the text window invokes the user’s default text editor. When a new comment is to be generated, a template appears already seeded with the name of the part being welded, the time segment to which the annotation applies, and hyperlinks to related graphs, images and associated comments. The user merely types his comment and saves the document, while the system takes care of naming the file. Annotations are stored on the local client until the user decides to submit them for group availability.

Benefits of Design

There are several advantages to structuring a tool using the architecture described above.

Distributed Control

Each tool is controlled by a separate AGNI agent that implements its driver. The driver reacts to events that can be generated from anywhere in the distributed application. For example, the slider tool can append messages to the controller that re-distributes these events as seek events to each of the tool drivers. If the multimedia tools support random seeks, they can respond to such seek requests and position their media appropriately, thereby giving the ability to have both real-time and manually controlled synchronization. If we wanted to share the slider, in a synchronous, collaborative fashion, the seek input simply needs to originate from another machine rather than the local slider. The control inputs could also come from another collaborative environment and indeed we have used this approach to integrate the tool with the Teamwave Workplace client.

Isolation of Components

Each tool runs in its own address space. Thus, a misbehaving tool cannot bring down the application. Failures are easy to isolate and fix. We can utilize off-the-shelf tools for media handling and annotation whenever such tools are available.

Modularity and Extensibility

Since all drivers export uniform interfaces, it is easy to add support for new media types. We simply build a driver to encapsulate the interface to the tool and plug it into the bus.

Performance

One of the future generalizations we envision for SMAT is the handling of real-time or continuous-time media such as audio and video. However, the tool also has to support annotations and still images that appear at discrete time intervals in the multimedia replay. To handle both media types, we have added a real-time clock input that sends a signal out on the control bus at a fixed, periodic interval. This signal is ignored by the continuous-time media players because they incorporate their own notion of time, but is interpreted by the discrete-time media players to initiate actions such as bringing up time-anchored annotations, advancing the cursor on a graph, etc. The smaller the interval of the clock tick, the finer the granularity of these discrete actions with respect to the

continuous-time media replay. In order for the tool as a whole to keep up with the clock rate of the bus, the tick has to propagate through the event bus and the discrete time media players have to complete their actions before the next tick.

To quantify how well this scheme worked, we measured the ability of SMAT to keep up with the clock ticks that are generated by the timer. By multiplying the tick interval by the number of ticks, we calculate what the wall clock time should have been provided the tool could keep up with the timer. This enables us to compute the average error, that is, the difference between the rendering rate of SMAT and the capture rate of the original dataset. We measured this error by letting the tool run for 30 s on a 130 MHz machine and tracking the number of ticks issued. (We used the configuration for SMAT shown in the GUI screen shot – Figure 6).

As shown in Figure 7, when the 30 s weld sequence was

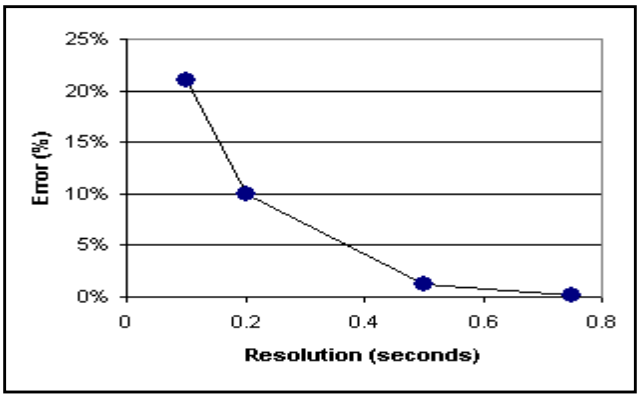

Figure 7: Effect of data sampling rate on deviation from real-time capture rate (130 MHz CPU)

broken into 0.75 s time slices, events were handled fast enough that the display was updated at a rate equivalent to the real time in which the data were originally collected. However, when the 30 s interval was broken into 300 slices of 0.1 s each, a significant lag was introduced into the event handling rate which caused a 20% slowdown in the rendering rate compared to real-time capture rate of the dataset. The results can be expected to improve using more contemporary hardware.

Initial Analysis of Use

Welding engineers and guest researchers recently participated in a welding exercise at NIST. An experiment was set up to test whether collaborative technologies could be effectively incorporated into this pseudo-manufacturing environment. Of the six participants in the experimental scenario, only two performed roles that involved the use of SMAT for weld evaluation. Due to the small number of users, the evaluation of SMAT is heuristic in nature and includes mainly measures of user satisfaction. The user-centered design methodology employed in developing the collaborative environment, including SMAT, has allowed several rounds of testing and prototype refinement. Observation and elicitation of talk aloud protocols were the main methods for capturing user feedback. The overall reaction to SMAT has been quite positive. Both users of the software believed that it was unique, providing capabilities that would be useful enough to incorporate into other tools that they currently use for reviewing weld quality. They recommended that additional types of data, such as static controller parameters, be added so that SMAT presents a more complete picture of the real-time context. The fact that SMAT displays could be invoked from within their normal collaborative system was met with enthusiasm since as one user noted, “All my work is in one place”, and “It uses common ways of accessing all my data.”

Future Development Directions

Since SMAT is a prototype tool that was developed employing user-centered design for a specific audience, its current features and capabilities have been targeted to address a specific set of requirements. To generalize the applicability of the tool, there are many potential enhancements that could be pursued. Among them are:

Real-time collaboration

SMAT could be enhanced to support real-time collaboration. Using this capability, users would effectively have partial control over each other’s tools to share the same view of the multimedia data. This could be accomplished by using the bus architecture and distributing control of one session to other sessions, as mentioned in the Benefits of Design section. Sharing display events is a logical next enhancement, but some thought and research is needed to determine the best candidate set of control events to share across multiple sessions.

Changing requirements at the desktop

It is envisioned that one very useful enhancement would be to modify SMAT so that more display option changes could be made at the desktop. Currently, a general set of options is provided at tool start-up. More application-specific choices could be made available to the user at start-up. Additionally, it is also possible to provide the user with the capability to make some display configuration changes during execution rather than just at start-up.

Handling of additional media types

As mentioned earlier, the infrastructure is present to add additional media players for new media types. Our prototype currently does not support video and audio explicitly, however, we do not expect adding support for additional media types to be a concern, merely an exercise.

Intelligent merging of annotations

SMAT supports disconnected operations. This means that there could be a scenario where more than one user decides to annotate the experiment at the same time offset from the start of the data. Currently, we do not support this level of annotation merging, and this is clearly a feature that is needed to make the tool robust.

Conclusion

We described the design and use of SMAT, a tool designed to be part of a scientific collaboratory for use in a robotic, arc-welding research project at the National Institute of Standards and Technology. SMAT met its primary functional requirements, providing the capability to capture, synchronize, play back, and annotate multimedia data in a multi-platform, distributed environment. In addition to describing SMAT's design, interface and underlying middleware, we presented performance information, an initial analysis of welding users' experiences and feedback, related work, and directions for further SMAT development.

References

[1] Abowd, G., "Software Engineering Issues for Ubiquitous Computing", Proceedings of the 21st International Conference on Software Engineering, Los Angeles, CA, 1999, pp. 75-84.

[2] Abowd, G., C. Atkeson, A. Feinstein, C. Hmelo, R. Kooper, S. Long, N. Sawhney and M. Tani, "Teaching and Learning as Multimedia Authoring: the Classroom 2000 Project", Proceedings of Multimedia 96, Boston, MA, 1996, pp. 187-198.

[3] Abowd, G., J. Brotherton and J. Bhalodia, "Classroom 2000: A System for Capturing and Accessing Multimedia Classroom Experiences", Proceedings of CHI 98, Los Angeles, CA, 1998, pp. 20-22.

[4] Bargeron, D., A. Gupta, J. Grudin and E. Sanocki, "Annotations for Streaming Video on the Web", Proceedings of CHI 99, Pittsburgh, PA, 1999,pp. 278-279.

[5] Box, D, *Essential COM*, Addison-Wesley Publishing Company, Menlo Park, California, 1998.

[6] Cheyer, A. and L. Julia, "MVIEWS: Multimodal Tools for the Video Analyst", Proceedings of the International Conference on Intelligent User Interfaces, San Francisco, CA, 1998, pp. 55-62.

[7] Greenberg, S., and M. Roseman, "Using a Room Metaphor to Ease Transitions in Groupware", Research Report 98/611/02, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, 1998.

[8] Landauer, T., *The Trouble with Computers: Usefulness, Usability and Productivity*, MIT Press, Cambridge, MA, 1995.

[9] Marcus, A. and A. van Dam, "User Interface Developments for the Nineties", Computer, IEEE, 1991, Vol. 249, pp.49-57.

[10] Nielsen, J., *Usability Engineering*, Academic Press, Boston, MA, 1993.

[11] Menon, J., "Collaborative Visualization and Modeling", Proceedings of International Conference on Shape Modeling and Applications, Aizu-Wakamatsu, Japan, 1997, pp.178-187.

[12] Ousterhout, J, *Tcl and the Tk Toolkit*, Addison-Wesley Publishing Company, Reading, MA, 1994.

[13] Pimentel, M.d. G., G.D. Abowd and Y. Ishiguro, "Linking by Interacting: A Paradigm for Authoring Hypertext", Proceedings of Hypertext 2000, Austin, TX, 2000, pp. 39-48.

[14] Ranganathan, M., M. Bednarek, F. Pors and D. Montgomery, "AGNI: A Multithreaded Middleware for Distributed Systems", 7th USENIX Tcl/Tk Conference, Austin, TX, 2000, pp. 11-21.

[15] Ranganathan, M., M. Bednarek and Doug Montgomery, "Reliable Communication for Mobile Agents", Proceedings of Agent Systems and Architectures/Mobile Agents (ASA/MA), Zurich, Switzerland, 2000, pp. 206-220.

[16] Rippey, W. and J. Falco, "The NIST Automated Arc Welding Testbed", Proceedings of the Seventh International Conference on Computer Technology in Welding, San Francisco, CA, 1997, pp. 203-210.

[17] Roseman, 2000, http://www.teamwave.com/.

[18] Schilit, B., G. Golovchinsky, and M. Price, "Beyond Paper: Supporting Active Reading with Free Form Digital Ink Annotations", Proceedings of CHI 98, Los Angeles, CA, 1998, pp. 249-256.

[19] Steves, M. and A. Knutilla, "Collaboration Technologies for Global Manufacturing", Proceedings of the ASME International Mechanical Engineering Congress and Exposition (IMECE): Symposium on Manufacturing Logistics in a Global Economy, Nashville, TN, 1999, pp. 541-555.

[20] Steves, M., W. Chang, and A. Knutilla, "Supporting Manufacturing Process Analysis and Trouble Shooting with ACT", Proceedings of the IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Stanford, CA, 1999, pp. 126-131.

[21] Williams, G., "Task Conflict and Language Differences: Opportunities for Videoconferencing?" Proceedings of the European Computer Supported Cooperative Work Conference (ECSCW), 1997, pp. 97-108.